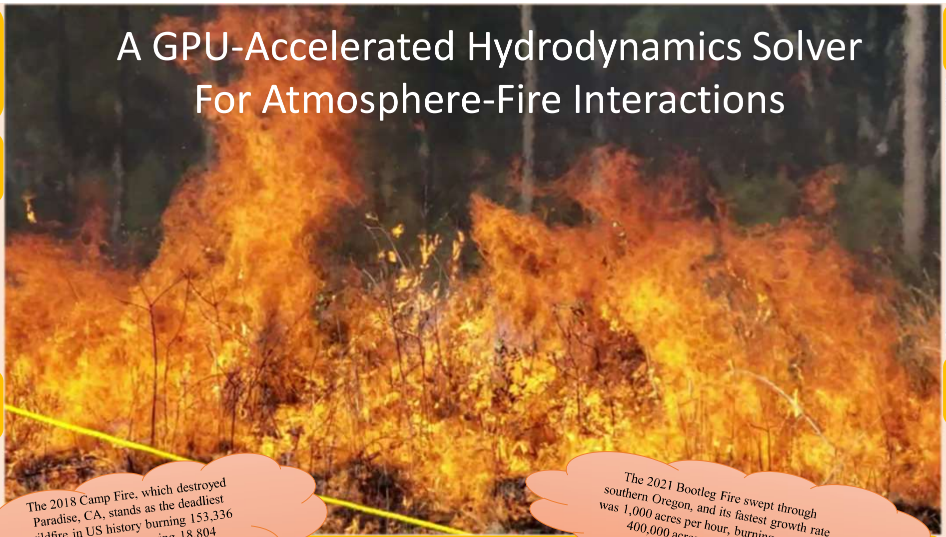
Jhamieka Greenwood jsg15h@my.fsu.edu
Advisors: Dr. Bryan Quaife and Dr. Kevin Speer
Department of Scientific Computing
Geophysical Fluid Dynamics Institute

A GPU-Accelerated Hydrodynamics Solver For Atmosphere-Fire Interactions



The 2018 Camp Fire, which destroyed Paradise, CA, stands as the deadliest wildfire in US history burning 153,336 acres of land, destroying 18,804 structures, costing over 16 billion dollars, and resulting in 85 fatalities.

The 2021 Bootleg Fire swept through southern Oregon, and its fastest growth rate was 1,000 acres per hour, burning more than 400,000 acres within two weeks.

Abstract

A fundamental process to understand fire spread is the atmospheric flow. Building computational tools to simulate this complex flow has several challenges including boundary layer effects, resolving vegetation and the forest canopies, conserving fluid mass, and incorporating fire-induced flows. We develop a two-dimensional hydrodynamic solver that models fire-induced flow as a convective sink that converts the two-dimensional horizontal flow into a vertical flow through the buoyant plume. The resulting equations are the two-dimensional Navier-Stokes equations, but with point source delta functions appearing in the conservation of mass equation. We develop a projection method to solve these equations and implement them on a GPU architecture. The ultimate goal is to simulate wildfire spread faster than real-time, and with the ability for users to introduce real-time updates in an augmented reality sandbox.

Methods

By studying scaled-down models for fire dynamics, we are investigating fundamental processes that are critical to fire spread. One of the most important factors is the complex coupling of the atmospheric flow and the combusting environment. In particular, what variables influence fire induced flows, and how do they contribute to fire spread? On the technology side, we are developing physics-based hydrodynamic models with a high degree of data parallelism that is ideal for graphical processing units (GPUs). The dynamics are visualized using OpenGL in an augmented reality environment.

Our hydrodynamic solver is based on the modifications of the Navier Stokes equations

$$\nabla\cdot\boldsymbol{u} = \begin{cases} -1, \boldsymbol{x}\ ignition\ site\ (\boldsymbol{x}_0) \\ 0, otherwise \end{cases} = -\delta(\boldsymbol{x}-\boldsymbol{x}_0) \qquad (1)$$

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \frac{1}{\rho}\boldsymbol{f} \qquad (2)$$

As an extension of the GPU-accelerated hydrodynamic solver of Stam [1], my preliminary work solves equation (1) using operator splitting and pressure projection. The projection step takes a velocity field $\boldsymbol{w}$ and finds the closet velocity field that satisfies the mass equation in equation (1).

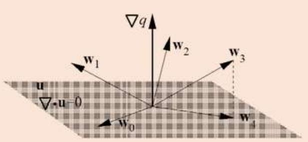


Figure 1: One simulation step of our solver is composed of steps. The first three steps may take the field out of the space of divergent free fields. The last projection step ensures that the field is divergent free after the entire simulation step.

Following the classical projection operator, the modification that we devised is

$$P\boldsymbol{w} = \boldsymbol{w} - \nabla q$$

$$\nabla^2 q = \nabla\cdot\boldsymbol{w}$$

$$where\ \nabla q = \nabla\cdot\boldsymbol{w} + \delta(\boldsymbol{x}-\boldsymbol{x}_0)$$

Implementation

addForce ( )

$$\boldsymbol{w}_1(\boldsymbol{x}) = \boldsymbol{w}_0(\boldsymbol{x}) + \Delta t\boldsymbol{f}(\boldsymbol{x},t)$$

Takes in an allocated device field and applies a force through user manipulation from the mouse. $\boldsymbol{f}$ is the force calculated from the motion coordinates of the mouse position multiplied by force scale factor, Δt, and force fall function $(\frac{1}{1+x^4+y^4})$.

advectVelocity ( )

$$\boldsymbol{w}_2(\boldsymbol{x}) = \boldsymbol{w}_1(\boldsymbol{p}(\boldsymbol{x},-\Delta t))$$

Takes in the device field from the addForce( ) routine and applies a semi-lagrangian method by performing a bilinear interpolation of the particle location (x,y) tracked backwards in time. The new velocity is the velocity the particle had $-\Delta t$ ago at the old location. The velocity field is now separated into two fields vxfield and vyfield for the next routine.

diffuseProject ( )

$$\widehat{\boldsymbol{w}_2} = FFT\{\boldsymbol{w}_2\} \qquad \widehat{\boldsymbol{w}_4}(\boldsymbol{k}) = \widehat{P}\widehat{\boldsymbol{w}_3} - \delta(\widehat{\boldsymbol{x}-\boldsymbol{x}_0})$$

$$\widehat{\boldsymbol{w}_3}(\boldsymbol{k}) = \frac{\widehat{\boldsymbol{w}_2}(\boldsymbol{k})}{1+\vartheta\Delta t\boldsymbol{k}^2} \qquad \boldsymbol{w}_4 = FFT^{-1}\{\widehat{\boldsymbol{w}_4}\}$$

The velocity fields, vxfield and vyfield, will be used in the diffuseProject( ) for operations on the real components of vxfield and vyfield (each having their own x and y components for FFT). Modifications from the previous slide are applied at the projection step. Inverse Fourier transform applied to the new fields.

updateVelocity ( )

The newly projected fields are normalized and combined back to the device field.

addSink ( )

$$\boldsymbol{w}_5(\boldsymbol{x}) = \boldsymbol{w}_4(\boldsymbol{x}) - \frac{1}{2\pi}\left(\frac{(\boldsymbol{x}-\boldsymbol{x}_0)}{|\boldsymbol{x}-\boldsymbol{x}_0|^2+\varepsilon^2}\right)$$

Convective sinks are added to the velocity field. Regularization is used to smooth out the singularity at x_0.

advectParticles ( )

$$\boldsymbol{part}(t+1) = \boldsymbol{part}(t) + dt * \boldsymbol{w}_5(\boldsymbol{part}(t))$$

The final routine takes the device field as a parameter and updates the particles by moving positions according to the velocity field and time. Then each particle is mapped to a texture object and bound for visualization.

Methods (cont.)

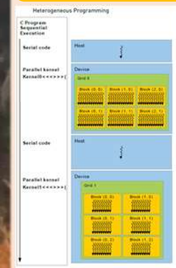


Figure 2. Heterogenous Programming Scheme

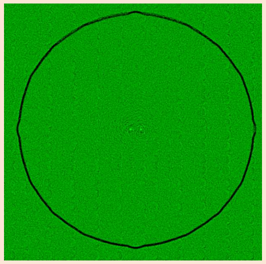
GPGPU Programming

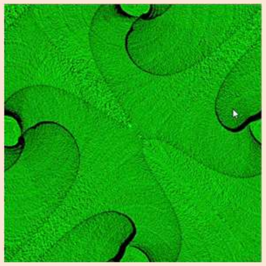
CUDA (Compute Unified Device Architecture) is a parallel computing technology developed by Nvidia for graphic processing. CUDA programming environment allow the development of general-purpose GPU programs for the implementation of numerical calculations on GPUs.

Heterogeneous Programming

CUDA programming model that consist of CUDA threads executing on a physically separate ***device*** (GPU) and operating as a coprocessor to the ***host*** (CPU) running the C++ program.

Results





Future Work

Interactivity will be incorporated with a 5×4 foot sand table filled with reflective sand. A projector displays output visuals from the connected computer onto the sand, and a Kinect camera determines the topography. The simulation will be coupled with a topographical model to visualize point-source spot fires spreading across a terrain and adding various variables. Hand gestures/motions will manipulate the simulation and the terrain can be altered in real-time by moving the structure of the sand

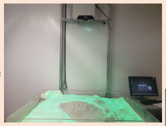


Figure 3. The photo to the right illustrates a topographical simulation projected onto reflective sand.

Figure 4. The photo to the left illustrates the augmented reality sand table running the fluid simulation. The dark regions are vortices in the flow.

References & Acknowledgements

[1]. Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 121–128, 1999.

Acknowledgements:
Florida State University, Department of Scientific Computing
Dr. Bryan Quaife, Dr. Kevin Speer